

Parameterized Compilation Lower Bounds for Restricted CNF-formulas

Stefan Mengel
CNRS, CRIL UMR 8288, France

April 25, 2016

We show unconditional parameterized lower bounds in the area of knowledge compilation, more specifically on the size of circuits in decomposable negation normal form (DNNF) that encode CNF-formulas restricted by several graph width measures. In particular, we show that

- there are CNF formulas of size n and modular incidence treewidth k whose smallest DNNF-encoding has size $n^{\Omega(k)}$, and
- there are CNF formulas of size n and incidence neighborhood diversity k whose smallest DNNF-encoding has size $n^{\Omega(\sqrt{k})}$.

These results complement recent upper bounds for compiling CNF into DNNF and strengthen—quantitatively and qualitatively—known conditional lower bounds for cliquewidth. Moreover, they show that, unlike for many graph problems, the parameters considered here behave significantly differently from treewidth.

1. Introduction

Knowledge compilation is a preprocessing regime that aims to translate or “compile” knowledge bases, generally encoded as CNF formulas, into different representations more convenient for a task at hand. The idea is that many queries one would like to answer on the knowledge base, say clause entailment queries, are intractable in CNF encoding, but tractable for other representations. When there are many queries on the same knowledge base, as for example in product configuration, it makes sense to invest into a costly preprocessing to change the representation once in order to then speed up the queries and thus amortize the time spent on the preprocessing.

One critical question when following this approach is the choice of the representation that the knowledge is encoded into. In general, there is a trade-off between the usefulness of a representation (which queries does it support efficiently?) and succinctness

(what is the size of the encoded knowledge base?). This trade-off has been studied systematically [6], leading to a fine understanding of the different representations. In particular, circuits in decomposable negation normal form (short DNNF) [7] have been identified as a representation that is more succinct than nearly all other representations while still allowing useful queries. Consequently, DNNFs play a central role in knowledge compilation.

This paper should be seen as complementing the findings of [1]: In that paper, algorithms compiling CNF formulas with restricted underlying graph structure were presented, showing that popular graph width measures like treewidth and cliquewidth can be used in knowledge compilation. More specifically, every CNF formula of incidence *treewidth* k and size n can be compiled into a DNNF of size $2^{O(k)}n$. Moreover, if k is the incidence *cliquewidth*, the size bound on the encoding becomes $n^{O(k)}$. As has long been observed, $2^{O(k)}n$ is of course far preferable to $n^{O(k)}$ for nontrivial sizes of n —in fact, this is the main premise of the field of parameterized complexity theory, see e.g. [14]. Consequently, the results of [1] leave open the question if the algorithm for clique-width based compilation of CNF formulas can be improved.

In fact, the paper [1] already gives a partial answer to this question, proving that there is no compilation algorithm achieving fixed-parameter compilability, i.e., a size bound of $f(k)p(|F|)$ for a function f and a polynomial p . But unfortunately this result is based on the plausible but rather non-standard complexity assumption that not all problem in $W[1]$ have FPT-size circuits. The result of this paper is that this assumption is not necessary. We prove a lower bound of $|F|^{\Omega(k)}$ for formulas of *modular incidence treewidth* k where modular treewidth is a restriction of cliquewidth proposed in [21]. It follows that the result in [1] is essentially tight. Moreover, we show a lower bound of $|F|^{\Omega(\sqrt{k})}$ for formulas of neighborhood diversity k [18]. This intuitively shows that all graph width measures that are stable under adding modules, i.e., adding a new vertex that has exactly the same neighborhood as an existing vertex, behave qualitatively worse than treewidth for compilation into DNNFs.

Related work. Parameterized knowledge compilation was first introduced by Chen [4] and has seen some recent renewed interest, see e.g. [5, 10] for work on conditional lower bounds. Unconditional lower bounds based on treewidth can e.g. be found in [22, 23], but they are only for different versions of branching programs that are known to be less succinct than DNNF. Moreover, these lower bounds fail for DNNFs as witnessed by the upper bounds of [1].

There is a long line of research using graph and hypergraph width measures for problems related to propositional satisfiability, see e.g. the extensive discussion in [3]. The paper [20] gave the first parameterized lower bounds on SAT with respect to graph width measures, in particular cliquewidth. This result was later improved to modular treewidth to complement an upper bound for model counting [21] and very recently to neighborhood diversity [11], a width measure introduced in [18]. We remark that the latter result could be turned into a conditional parameterized lower bound similar to that in [1] discussed above.

Our lower bounds strongly rely on the framework for DNNF lower bounds proposed

in [2] and communication theory lower bounds from [13], for more details see Section 3.

2. Preliminaries

In the scope of this paper, a *linear code* C is the solution of a system of linear equations $A\bar{x} = 0$ over the boolean field \mathbb{F}_2 . The matrix A is called the *parity-check matrix* of C . The characteristic function f_C is the boolean function that, given a boolean string e , evaluates to 1 if and only if e is in C .

We use the notation $[n] := \{1, \dots, n\}$ and $[n_1, n_2] := \{n_1, n_1 + 1, \dots, n_2\}$ to denote integer intervals. We use standard notations from graph theory and assume the reader to have a basic background in the area [12]. By $N(v)$ we denote the open neighborhood of a vertex in a graph.

We say that two vertices u, v in a graph $G = (V, E)$ have the same neighborhood type if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. It can be shown that having the same neighborhood type is an equivalence relation on V [18]. The neighborhood diversity of G is defined to be the number of equivalence classes of V with respect to neighborhood types.

A generalization of neighborhood diversity is *modular treewidth* which is defined as follows: From a graph G we construct a new graph G' by contracting all vertices sharing a neighborhood type, i.e., from every equivalence class we delete all vertices but one. The modular treewidth of G is then defined to be the treewidth of G' ¹. Modular pathwidth is defined in the obvious analogous way.

We assume basic familiarity with propositional logic and in particular CNF formulas. We define the size of a CNF formula to be the overall number of occurrences of literals, i.e., the sum of the sizes of the clauses where the size of a clause is the number of literals in the clause. The incidence graph of a CNF formula F has as vertices the variables and clauses of F and an edge between every clause and the vertices contained in it. The projection of an assignment $a : X \rightarrow \{0, 1\}$ to a set Z is the restriction of a to the variable set Z . This definition generalizes to sets of assignments in the obvious way. Moreover, the projection of a boolean function f on X to Z is defined as the boolean function on Z whose satisfying assignments are those of f projected to Z . For the width measures introduced above, we define the width of a formula to be that of its incidence graph.

3. Statement of the Main Results and Preparation of the Proof

We now state our main results. The first theorem shows that modular pathwidth—and thus also more general parameters like cliquewidth and modular treewidth—do not allow fixed-parameter compilation to DNNF.

¹Note that the definition in [21] differs from the one we give here, but can easily be seen to be equivalent for bipartite graphs and thus incidence graphs of CNF formulas. We keep our definition to be more consistent with the definition of neighborhood diversity.

Theorem 1. *For every k and for every n big enough there is a CNF formula F of size at most n and modular pathwidth k such that any DNNF computing the same function as F must have size $n^{\Omega(k)}$.*

We also show lower bounds for neighborhood diversity that are nearly as strong as those for modular pathwidth.

Theorem 2. *For every k and for every n big enough there is a CNF formula F of size polynomial in n and with neighborhood diversity k such that any DNNF computing the same function as F must have size $n^{\Omega(\sqrt{k})}$.*

At this point, the attentive reader may be a little concerned because we promise to prove lower bounds for DNNF which we have not even defined in the preliminaries. In fact, it is the main strength of the approach in [2] that the definition and properties of DNNF are not necessary to show our lower bounds, because we can reduce showing lower bounds on DNNF to a problem in communication complexity. Since we will not use any properties of DNNF, we have decided to leave out the definition out of the main text; the interested reader may find a short overview in Appendix A. Here we will only use the following result.

Theorem 3 ([2]). *Let f be a function computed by a DNNF of size s . Then f has a multi-partition rectangle cover of size s .*

Now the reader might be a little puzzled about what multi-partition rectangle covers of a function are. Since we will also only use them as a black box in our proofs and do not rely on any of their properties, we have opted to leave out their definition the main text. The curious reader may find a very short introduction into the beautiful area of multi-partition communication complexity in Appendix B.

We will use a powerful theorem which follows directly from the results in [13].

Theorem 4 ([13]). *For every $n' \in \mathbb{N}$ and every $m' \leq n'/32$ there is a linear code C with a $m' \times n'$ parity check matrix such that every multi-partition rectangle cover of the characteristic function f_C has size at least $\frac{1}{4}2^{m'}$.*

4. Accepting Codes by CNF Formulas

In this section we will construct CNF formulas to accept linear codes. We will first start with a naive encoding that will turn out to be of unbounded modular treewidth and thus not directly helpful to us. We will then show how to change the encoding in such a way that the modular treewidth and even the neighborhood diversity are small and the size of the resulting CNF is small enough to show meaningful lower bounds for encodings in DNNF with Theorem 4.

4.1. The naive approach

In this subsection, we show how we can check m linear equations on variables x_1, \dots, x_n efficiently by CNF. The idea is to simply consider one variable after the other and

remember the parity for the equations at hand. To this end, fix an $m \times n$ matrix $A = (a_{ij})$. We denote the resulting equations of the system $A\bar{x} = 0$ by E_1, \dots, E_m . For each equation E_i we introduce variables z_{ij} for $j \in [n]$ which intuitively remembers the parity of E_i up to seeing the variable x_j .

We encode the computations for each E_i individually: Introduce constraints

$$a_{i,1}x_1 = z_{i,1}, \quad (1)$$

$$z_{i,j-1} + a_{ij}x_j = z_{ij}. \quad (2)$$

Note that $z_{i,n}$ yields the parity for equation E_i which can then be checked for 0. This yields a system whose accepted inputs projected to the x_i are the code words of the considered code. The constraints have all at most 3 variables, so we can encode them into CNF easily.

Unfortunately, the resulting CNF can be shown to have high modular treewidth, so it is not useful for our considerations. We will see how to reduce the modular treewidth and the neighborhood diversity of the system without blowing up the size of the resulting CNF-encoding too much.

4.2. Bounding modular treewidth

The idea for decreasing the modular treewidth is to not encode all constraints on the parities individually but combine them into larger constraints. So fix n and k and set $m := k \log(n)$. For each j , we will combine the constraints from (1) and (2) for blocks of $\log(n)$ values of i into one. The resulting constraints are

$$R_1^\ell(x_1, z_{\ell \log(n)+1,1}, \dots, z_{(\ell+1) \log(n),1}) := \{(d_1, t_{\ell \log(n)+1,1}, \dots, t_{(\ell+1) \log(n),1}) \mid a_{i,1}d_1 = t_{i,1}, i = \ell \log(n) + 1, \dots, (\ell+1) \log(n)\}$$

and

$$R_j^\ell(x_i, z_{\ell \log(n)+1,j-1}, \dots, z_{(\ell+1) \log(n),j-1}, z_{\ell \log(n)+1,j}, \dots, z_{(\ell+1) \log(n),j}) := \{(d_i, t_{\ell \log(n)+1,j-1}, \dots, t_{(\ell+1) \log(n),j-1}, t_{\ell \log(n)+1,j}, \dots, t_{(\ell+1) \log(n),j}) \mid t_{i,j-1} + a_{ij}d_j = t_{i,j}, i = \ell \log(n) + 1, \dots, (\ell+1) \log(n)\}$$

for $\ell = 0, \dots, k-1$.

Note that the constraints R_j^ℓ have at most $2 \log(n) + 1$ boolean variables, so we can encode them into CNF of quadratic size where every clause contains all variables of R_j^ℓ . Moreover, the R_j^ℓ encode all previous constraints from (1) and (2), so the assignments satisfying all R_j^ℓ projected to the x_i still are exactly the code words of the code we consider. Call the resulting CNF F .

Claim 1. F has modular pathwidth at most $2k - 1$.

Proof. Note that the clauses introduced when translating the constraint R_j^ℓ into CNF have by construction all the same set of variables. Thus these clauses have the same

neighborhood type, and we can for modular treewidth restrict to an instance just having one clause for each R_j^ℓ . We call the resulting vertex in the incidence graph $r_{\ell,j}$. Next, observe that the variables $z_{\ell \log(n)+i,j}$ and $z_{\ell \log(n)+i',j}$ for $i, i' \in [\log(n)]$ appear in exactly the same clauses. Thus these variables have the same neighborhood type as well, so we can delete all but one of them, say $z_{\ell \log(n),j}$ for $\ell = 1, \dots, k$. Call the resulting vertices in the incidence graph $s_{\ell,j}$.

The resulting graph $G = (V, E)$ has

$$\begin{aligned} V &= \{x_j, s_{\ell,j}, r_{\ell,j} \mid j \in [n], \ell \in [k]\}. \\ E &= \{x_j r_{\ell,j} \mid j \in [n], \ell \in [k]\} \cup \{s_{\ell,j-1} r_{\ell,j} \mid j \in [2, n], \ell \in [k]\} \\ &\quad \cup \{s_{\ell,j} r_{\ell,j} \mid j \in [n], \ell \in [k]\} \end{aligned}$$

We construct a path decomposition of G as follows: The bags are the sets

$$\begin{aligned} B_2 &:= \{x_1 r_{1,1}, \dots, r_{\ell,1}\}, \\ B_3 &:= \{s_{1,1}, \dots, s_{\ell,1}, r_{1,1}, \dots, r_{\ell,1}\} \end{aligned}$$

and for $j = 2, \dots, n$

$$\begin{aligned} B_{3j-2} &:= \{s_{1,j-1}, \dots, s_{\ell,j-1}, r_{1,j}, \dots, r_{\ell,j}\}, \\ B_{3j-1} &:= \{x_j, r_{1,j}, \dots, r_{\ell,j}\}, \text{ and} \\ B_{3j} &:= \{s_{1,j}, \dots, s_{\ell,j}, r_{1,j}, \dots, r_{\ell,j}\}. \end{aligned}$$

Ordering the bags B_j with respect to their index yields a path decomposition of G of width $2k - 1$. \square

Let us collect the results of the section into one statement.

Lemma 1. *For every linear code C with a $k \log(n) \times n$ parity check matrix there is a CNF formula F in variable sets X and Z such that*

- *the solution set of F projected to X is exactly C ,*
- *F has size $O(kn^3 \log(n)^2)$, and*
- *F has modular pathwidth at most $2k - 1$.*

Proof. It remains only to show the size bound on F . Note that we have n variables x_j and $kn \log(n)$ variables $z_{i,j}$. Moreover, we have $kn \log(n)$ constraints R_i^ℓ . Each of those has $2 \log(n) + 1$ variables, so it can be encoded by $O(n^2)$ clauses with $O(\log(n))$ variables each. This yields $O(kn^3 \log(n))$ clauses with $O(\log(n))$ variables. Consequently, the overall size of F is $O(kn^3 \log(n)^2)$. \square

4.3. Bounding neighborhood diversity

Fix now two positive integers N and k and let $n := 32k \log(N)$ and $m := k \log(N)$ and consider A with these parameters as before. We want to encode the code of A by a CNF with neighborhood diversity $O(k^2)$.

To do so, we split the variables z_{ij} into $O(k^2)$ sets of $\log(N)^2$ variables $S_{rs} := \{z_{ij} \mid i \in [r \log(N) + 1, (r+1) \log(N)], j \in [s \log(N) + 1, (s+1) \log(N) - 1]\}$ for $r \in [k]$ and $s \in [32k]$. Now create for all $r \in [k]$, $s \in [32k - 1]$ a constraint R_{rs} in the variables $X := \{x_1, \dots, x_n\} \cup S_{rs} \cup S_{r+1,s}$ that accepts all assignments to its variables that satisfy all constraints from 1 and 2 whose variables are variables of R_{rs} . Note that the resulting constraints cover all constraints of Section 4.1, so we still accept the code defined by A after projection to X .

The problem now is that, since we have $\Theta(\log(N)^2)$ boolean variables in each constraint, the resulting encoding into CNF could be superpolynomial in N and thus too big for our purposes. This is easily repaired by the observation that fixing the values of x_i , $z_{i,s \log(N)}$ and $z_{i,s \log(N)+1}$ determines the values of the other z_{ij} in all satisfying assignments. Consequently, we can project out these variables of the individual constraints without changing the accepted assignments to X . Call the resulting constraints R'_{rs} . It is easy to see that every constraint R'_{rs} has only $O(\log(N))$ variables, so the CNF encoding in which every variable of R'_{rs} appears in every clause has polynomial size in N .

We claim that the resulting CNF F has neighborhood diversity $O(k^2)$. To see this, note that the clauses introduced in the encoding of a fixed R'_{rs} all have the same variables. It follows that the clause vertices in the incidence graph have $O(k^2)$ neighborhood types. The variables in X all appear in all clauses, so they are all of the same neighborhood type. Finally, the vertices in each S_{rs} appearing in an R'_{rs} all appear in the same clauses, so they have $O(k^2)$ neighborhood types as well. This shows that the incidence graph of the CNF formula F has neighborhood diversity $O(k^2)$.

Let us again combine the results of this section into one summary statement.

Lemma 2. *For every linear code C with a $k \log(N) \times 32k \log(N)$ parity check matrix there is a CNF formula F in variable sets X and Z such that*

- *the solution set of F projected to X is exactly C ,*
- *F has size polynomial in N and k , and*
- *F has neighborhood diversity $O(k^2)$.*

5. Completing the Proof

We now combine the results of the previous sections to get our main results.

of Theorem 1. Let C be a linear code as in Theorem 4 with parameters $n' = n^{\frac{1}{4}}$ and $m' := \log(n)^{\frac{k}{2}} = k \log(n^{\frac{1}{4}})$. Then by Theorem 4 we know that every rectangle cover of the characteristic function f_C has size at least $\frac{1}{4} 2^{m'} = n^{\Omega(k)}$.

Now apply Lemma 1 to C to get a CNF-formula F of size less than n and modular pathwidth less than k . Let D be a DNNF representation of F of minimal size s . Since DNNFs allow projection to a subset of variables without any increase of size [7], this yields a DNNF of size s computing f_C . But then by Theorem 3, we get that $s \geq n^{\Omega(k)}$. \square

With the same proof but other parameters we get Theorem 2 from Lemma 2.

6. Connections to Model Counting and Affine Decision Trees

In this section we discuss connections of the findings of this paper to practical model counting. It has been shown that there is a tight connection between compilation and model counting, as runs of exhaustive DPLL-based model counting algorithms can be translated into (restricted) DNNFs [15]. Here the size of the resulting DNNF corresponds to the runtime of the model counter. Since state of the art solvers like Cachet [25] and sharpSAT [26] use exhaustive DPLL, the lower bounds in this paper can be seen as lower bounds for these programs: model counting for CNF formulas of size n and, modular treewidth will take time at least $n^{\Omega(k)}$ when solved with these state-of-the-art solvers even with perfect caching and optimal branching variable choices. Note that in the light of the general conditional hardness result of [21] this is not surprising, but here we get concrete and unconditional lower bounds for a large class of algorithms used in practice. Naturally, we also directly get lower bounds for approaches that are based on compilation into DNNF as those in [8, 19], so we have lower bounds for most practical approaches to model counting.

One further interesting aspect to observe is that, while the instances that we consider are in a certain sense hard for practical model counting algorithms, in fact counting their models is extremely easy. Since we just want to count the number of solutions of a system of linear equations, basic linear algebra will do the job. A similar reasoning translated to compilation is the background for the definition of affine decision trees (ADT) [16], a compilation language that intuitively has checking an affine equation as a built-in primitive. Consequently, it is very easy to see that ADTs allow a very succinct compilation of the CNF formulas we consider in this paper. It follows, by setting the right parameters, that there are formulas where ADTs are exponentially more succinct than DNNF. We remark that this superior succinctness can also be observed in experiments when compiling the formulas of Section 4.1 with the compiler from [16].

7. Conclusion

We have shown that parameters like cliquewidth, modular treewidth and even neighborhood diversity behave significantly differently from treewidth for compilation into DNNF by giving lower bounds complementing the results of [1]. These unconditional

lower bounds confirm conditional ones that had been known for some time already and improve them quantitatively. Our proofs heavily relied on the framework proposed in [2] thus witnessing the strength of this approach. We have also discussed implications for practical model counting.

One consequence of our results is that most graph width measures that allow dense incidence graphs for the input CNF—like modular treewidth or cliquewidth and unlike treewidth which forces a small number of edges—do not allow fixed-parameter compilation into DNNF. A priori, there is no reason why many edges in the incidence graphs, which translates into many big clauses, should necessarily make compilation hard. Thus it would be interesting to see if there are any width measures that allow dense graphs and fixed-parameter compilation at the same time. One width measure that might be worthwhile analyzing is the recently defined measure *sm-width* [24].

Acknowledgments. The author would like to thank Florent Capelli for helpful discussions. Moreover, he thanks Jean-Marie Lagniez for helpful discussions and for experiments with the compiler from [16].

References

- [1] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. On Compiling CNFs into Structured Deterministic DNNFs. In Marijn Heule and Sean Weaver, editors, *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, volume 9340 of *Lecture Notes in Computer Science*, pages 199–214. Springer, 2015.
- [2] Simone Bova, Florent Capelli, Stefan Mengel, and Friedrich Slivovsky. Knowledge Compilation Meets Communication Complexity. In *IJCAI 2016, Proceedings of the 25th International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2016. to appear.
- [3] Johann Brault-Baron, Florent Capelli, and Stefan Mengel. Understanding Model Counting for beta-acyclic CNF-formulas. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPICs*, pages 143–156. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
- [4] Hubie Chen. Parameterized Compilability. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 412–417. Professional Book Center, 2005.
- [5] Hubie Chen. Parameter Compilation. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPICs*, pages 127–137. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

- [6] A. Darwiche and P. Marquis. A Knowledge Compilation Map. *J. Artif. Intell. Res. (JAIR)*, 17:229–264, 2002.
- [7] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.
- [8] Adnan Darwiche. New Advances in Compiling CNF into Decomposable Negation Normal Form. In Ramon López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI’2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 328–332. IOS Press, 2004.
- [9] Adnan Darwiche. SDD: A New Canonical Representation of Propositional Knowledge Bases. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 819–826. IJCAI/AAAI, 2011.
- [10] Ronald de Haan. An Overview of Non-Uniform Parameterized Complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:130, 2015.
- [11] Holger Dell, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Tobias Mömke. Complexity and Approximability of Parameterized MAX-CSPs. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 294–306. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
- [12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- [13] Pavol Duris, Juraj Hromkovic, Stasys Jukna, Martin Sauerhoff, and Georg Schnitger. On multi-partition communication complexity. *Inf. Comput.*, 194(1):49–75, 2004.
- [14] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.
- [15] Jinbo Huang and Adnan Darwiche. DPLL with a Trace: From SAT to Knowledge Compilation. In Leslie Pack Kaelbling and Alessandro Saffioti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 156–162. Professional Book Center, 2005.
- [16] Frédéric Koriche, Jean-Marie Lagniez, Pierre Marquis, and Samuel Thomas. Knowledge Compilation for Model Counting: Affine Decision Trees. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013.

- [17] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
- [18] Michael Lampis. Algorithmic Meta-theorems for Restrictions of Treewidth. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, volume 6346 of *Lecture Notes in Computer Science*, pages 549–560. Springer, 2010.
- [19] Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-DNNF Compilation with sharpSAT. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence - 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, May 28-30, 2012. Proceedings*, volume 7310 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.
- [20] Sebastian Ordyniak, Daniël Paulusma, and Stefan Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. *Theor. Comput. Sci.*, 481:85–99, 2013.
- [21] Daniël Paulusma, Friedrich Slivovsky, and Stefan Szeider. Model Counting for CNF Formulas of Bounded Modular Treewidth. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, volume 20 of *LIPICs*, pages 55–66. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
- [22] Igor Razgon. No Small Nondeterministic Read-Once Branching Programs for CNFs of Bounded Treewidth. In Marek Cygan and Pinar Heggernes, editors, *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, volume 8894 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2014.
- [23] Igor Razgon. On OBDDs for CNFs of Bounded Treewidth. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014.
- [24] Sigve Hortemo Sæther and Jan Arne Telle. Between Treewidth and Clique-Width. In Dieter Kratsch and Ioan Todinca, editors, *Graph-Theoretic Concepts in Computer Science - 40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014. Revised Selected Papers*, volume 8747 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2014.
- [25] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, and Toniann Pitassi. Combining Component Caching and Clause Learning for Effective Model Counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

- [26] Marc Thurley. sharpSAT - Counting Models with Advanced Component Caching and Implicit BCP. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 424–429. Springer, 2006.

A. DNNF

In this section of the appendix, we give a short primer on DNNFs. For more background, we recommend the very influential paper [6].

A (*Boolean*) *circuit in negation normal form* (or *NNF*) is a directed acyclic graph (DAG) with a single sink node (outdegree 0) where each source node (indegree 0) is labelled by a constant (0 or 1) or by a literal, and each other node is labelled by \wedge (AND) or \vee (OR). If φ is an NNF and v is a vertex of φ , the *sub-NNF* of φ rooted at v is the NNF obtained from φ by deleting every vertex from which v cannot be reached along a directed path. We write $\text{var}(\varphi)$ for the set of variables occurring in an NNF φ . Let φ be an NNF and let τ be an assignment to $X \supseteq \text{var}(\varphi)$. Relative to τ , we associate each vertex v of φ with a value $\text{val}_\varphi(v, \tau) \in \{0, 1\}$ as follows. If v is labelled with a constant $c \in \{0, 1\}$ then $\text{val}_\varphi(v, \tau) = c$, and if v is labelled with a literal ℓ then $\text{val}_\varphi(v, \tau) = \tau(\ell)$. If v is an AND node then we let $\text{val}_\varphi(v, \tau) = \min\{\text{val}_\varphi(w, \tau) \mid w \text{ is a child of } v\}$, and if v is an OR node we define $\text{val}_\varphi(v, \tau) = \max\{\text{val}_\varphi(w, \tau) \mid w \text{ is a child of } v\}$. We say that τ *satisfies* φ if $\text{val}_\varphi(s, \tau) = 1$, where s denotes the (unique) sink of φ . The function computed by φ is defined in the obvious way.

An NNF φ is *decomposable* (in short, a *DNNF*) if every AND node v of φ satisfies the following property: if v has incoming edges from v_1 and v_2 , and φ_1 and φ_2 denote the sub-NNFs of φ rooted at v_1 and v_2 , respectively, then $\text{var}(\varphi_1)$ and $\text{var}(\varphi_2)$ are disjoint. A DNNF φ is *deterministic* (a *d-DNNF*) if, for every pair of distinct children v_1 and v_2 of an OR node, the sub-NNFs rooted at v_1 and v_2 do not have satisfying assignments in common.

DNNFs have first been defined by Darwiche [7] and since then played a central role in knowledge compilation. There are several reasons for this success: It has been shown [6] that DNNFs can be seen as a generalization of many other successful representations that are used in knowledge compilation, in particular classical languages like OBDDs and FBDDs. Note however that DNNFs are in general far more succinct than these classical representation. Seeing known compilation languages as subclasses of DNNFs has several advantages. In particular it facilitates the structured and systematic comparison of different representations. Moreover, this systematic understanding allows to easily define new compilation languages as subclasses of DNNFs that have desirable properties for a task at hand. For example, the class of sentential decision diagrams [9] is a subclass of DNNF defined in such a way as to have canonical representations, a property that often plays a central role in practice. Similarly, deterministic DNNFs are a restriction of DNNFs that is useful whenever efficient model counting is required. Arguably, seeing DNNFs as a unifying framework for the creation of compilation languages has been very influential in the field.

B. Some communication complexity

We introduce some very limited notions of communication complexity. The interested reader is referred to [17] for general background and to [13] for the rather specific notions and results on multi-partition communication complexity that appear in this paper.

Let f be a boolean function defined on a set X of n boolean variables. Let $\Pi = (X_1, X_2)$ be a partition of X . We say that Π is β -balanced for $\beta > 0$ if $\min(|X_1|, |X_2|) \geq \beta|X|$. The exact value of β is unsubstantial for most of our considerations, and it is most of the time only necessary to assume that there is some constant β for which all partitions that are considered are β -balanced. Consequently one often simply speaks of *balanced partitions*.

A *combinatorial rectangle* with respect to the partition Π is a function $r : \{0, 1\}^n \rightarrow \{0, 1\}$ that can be written as $r^{(1)} \wedge r^{(2)}$, where the functions $r^{(1)}, r^{(2)} : \{0, 1\}^n \rightarrow \{0, 1\}$ depend only on the variables in X_1 and X_2 , respectively. We define a (*multi-partition*) *rectangle cover* of f of size t to be a set of rectangles $\{r_1, \dots, r_t\}$ such that $f = r_1 \vee r_2 \vee \dots \vee r_t$. Note that in a rectangle cover each rectangle may be with respect to its own balanced partition of the variables in X .

Rectangle covers generalize similar notions of covers in communication complexity by allowing different partitions for the different rectangles. They were originally defined to prove lower bounds for different types of branching programs in which the variables are not met in a fixed order. In [1] it was shown that lower bounds on the size of rectangle covers also give lower bounds for different versions of DNNFs, see e.g. Theorem 3.